When you are working on project, you might want to go to last lean and working status. So basically you want to come back to the last commit in your branch but you kind of want to save your draft that you are working on. With **git stash** it is easily possible without having to create a new commit for this draft status. So you can avoid these dirty commits you could say so. So lets see how this works.

Let’s say we have file index.html, there we have one commit. code is-

<p>First Commit</p>

Now les say you are working on some feature and you have added some code. So you are working and not finished yet but you want to go your initial situation, i.e when you had only 1 commit in here. As I said in beginning, creating a new commit is possible but is a dirty approach, therefore you can use command . we have this code-

<p>First Commit</p>

<p>Feature1</p>

Now run –

**D:\gitStash>git stash**

**Saved working directory and index state WIP on master: be0c2f1 First Commit**

You are taken back to first commit. so content that we added is gone and we are back to first commit. run **git log** and you will see only 1 commit. note:- unstaged as well staged changes can be stashed.

Run-

**D:\gitStash>git stash apply**

**On branch master**

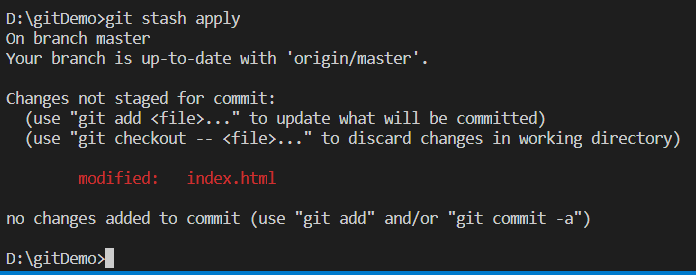
**Changes not staged for commit:**

**(use "git add <file>..." to update what will be committed)**

**(use "git checkout -- <file>..." to discard changes in working directory)**

**modified: index.html**

**no changes added to commit (use "git add" and/or "git commit -a")**



now we will get back code that we had before we ran git stash. This is because our stash status, our stash project status is loaded you could say, back again, into our main project. As you can see nothing is commited so far, you have option to commit this change and add it our project or to keep on working on this stash, so on this draft version. Lets say we are also working on feature2. So we have this file-

<p>First Commit</p>

<p>Feature1</p>

<p>Feature2</p>

run this-

**git stash**

**Saved working directory and index state WIP on master: be0c2f1 First Commit**

Now we are again back to code that we had when we had first commit. now run-

**D:\gitStash>git stash list**

**stash@{0}: WIP on master: be0c2f1 First Commit**

**stash@{1}: WIP on master: be0c2f1 First Commit**

so we have 2 stashes with index of 0 and 1 right here.

If we run –

**git stash apply**

**On branch master**

**Changes not staged for commit:**

**(use "git add <file>..." to update what will be committed)**

**(use "git checkout -- <file>..." to discard changes in working directory)**

**modified: index.html**

**no changes added to commit (use "git add" and/or "git commit -a")**

we can see that we have moved to last stash status. i.e to feature2.so last stash will have index 0. We can use index numbers with **git stash apply**  to select to select to which stash version we want to go .but we just run **git stash,** it will take you to most recent stash i.e top stash(stash {0})

If you now run-

**D:\gitStash>git stash apply 1**

**error: Your local changes to the following files would be overwritten by merge:**

**index.html**

**Please commit your changes or stash them before you merge.**

**Aborting**

This is not working because we applied a later stash, the top stash, the most recent stash(to be precise:(we previously applied the latest stash “stash@{0}, therefore applying the earlier “stash@{1}” does’nt work). So now we want to go stash{1}. To do this-

Run-

**git stash**

**D:\gitStash>git stash list**

**stash@{0}: WIP on master: be0c2f1 First Commit**

**stash@{1}: WIP on master: be0c2f1 First Commit**

**stash@{2}: WIP on master: be0c2f1 First Commit**

we just created stash 0 and stash 2(index just increased) is stash which we want to checkout. So run-

**git stash apply 2**

**On branch master**

**Changes not staged for commit:**

**(use "git add <file>..." to update what will be committed)**

**(use "git checkout -- <file>..." to discard changes in working directory)**

**modified: index.html**

**no changes added to commit (use "git add" and/or "git commit -a")**

now we are back at feature 1. Code-

index.html-

<p>First Commit</p>

<p>Feature1</p>

Lets edit this file.index,html-

<p>First Commit</p>

<p>Feature1</p>

<p>Another awe some feature</p>

Now we can run git stash, which will basically add stash on top of anothers or we can run –

**git stash push –m”description”**

we ran-

**D:\gitStash>git stash push -m "addedFeature"**

**Saved working directory and index state On master: addedFeature**

this description make it easier for us to identify the stash and what we changed there. above command creates stash of our changes . Now if we run-

**D:\gitStash>git stash list**

**stash@{0}: On master: addedFeature**

**stash@{1}: WIP on master: be0c2f1 First Commit**

**stash@{2}: WIP on master: be0c2f1 First Commit**

**stash@{3}: WIP on master: be0c2f1 First Commit**

**stash@{4}: WIP on master: be0c2f1 First Commit**

**stash@{5}: WIP on master: be0c2f1 First Commit**

we can see that we have description of last added stash. Now lets say I want to drop some stashes. Lets say we dnt need them. Run-

**D:\gitStash>git stash drop 3**

**Dropped refs/stash@{3} (8cd85f43ec5de1e5233cbc83bf0226335f70f77a)**

here 3 is index number of stash that we want to drop. We can also do one thing here. Lets say we want go implement a stash(i.e commit that code), then move out of stash by running stash(i.e move to main commit) and run-

**git stash pop 1**

**On branch master**

**Changes not staged for commit:**

**(use "git add <file>..." to update what will be committed)**

**(use "git checkout -- <file>..." to discard changes in working directory)**

**modified: index.html**

**no changes added to commit (use "git add" and/or "git commit -a")**

**Dropped refs/stash@{1} (84fbe00871ffaa3ff43d26900b6fa27ffe7ddcab)**

Now here we will get code of stash 1 and stash 1 will be deleted and index will be adjusted accordingly(its like array).

Now we can stage changes(we re in commit version not in stash) then we can commit them. now if we run **git log**, we will only see 2 commits. Now we have implemented a feature and I do not need stash anymore, then run –

**git stash clear**

now you can run, **git stash list** . you will see we do not have any stash versions of our code, saved right here.

So that is actually stash command , explained in few words.as you can see this is quite helpful if you want to implement new features in code and play around and you do not want to destroy actual working main code that you have in your branch and in last commit, for example.